Counting Essential Graphs

Gernot D. Kleiter *
Department of Psychology
University of Salzburg, Austria
gernot.kleiter@sbg.ac.at

Abstract

The contribution describes a method how to enumerate essential graphs. The method is based on two concepts, boundary vertices and symmetric vertices. Boundary vertices generalize the concept of terminal vertices to graphs with directed and undirected edges. Symmetric vertices are defined recursively as vertices that have symmetric neighbors. Classes of symmetric vertices allow to count labeled essential graphs. Layers of boundary vertices together with symmetry allow to find maximal essential graphs which are canonical representatives and can be used for counting unlabeled essential graph structures. Possible applications of the work are discussed. One goal is to enable the introduction of appropriate prior distributions in the model space for learning conditional independence structures.

1 Introduction

Conditional independence models—especially Bayesian networks—are the most successful and popular class of models to represent complex and uncertain knowledge. It is well known that the representation of conditional independence models by Bayesian networks is not unique in the sense that probabilistically equivalent models may have several different graphical representations. Essential graphs avoid this difficulty. They represent conditional independence models by graphs containing both directed and undirected edges.

In the present paper we describe methods to enumerate labeled and unlabeled essential graphs. To our knowledge this problem has not satisfactorily been solved. Important contributions were made by Gillispie and Perlman [3] and by Steinsky [14, 15].

Before going into details we explain why we think that enumerating essential graphs is important.

1. First of all, we are concerned with the enumeration of *essential graphs* and not of Bayesian networks for two reasons: (i) the number of Bayesian

*This research was supported by the Austrian-Czech AKTION Project.

networks (or equivalently, of directed acyclic graphs) with n variables is known [12] and (ii) Bayesian networks are completely inappropriate for counting conditional independence models because equivalent models are counted several times. Essential graphs were introduced [1] to avoid the problem of treating equivalent models as though they were different. Essential graphs do not lead to multiple counts.

2. How do we evaluate how well a model "fits" the data? Unfortunately, often *local* criteria are used for model evaluation. Typical local criteria are testing independence conditions at individual vertices in a Bayesian network. This leads to many tests with the same data set. The tests are not independent and this causes a breakdown of the traditional statistical testing principles.

   Model evaluation should be based on *holistic* criteria. A holistic criterion is, e.g., the posterior probability of a model given the data, $P(M_i|D)$. To work with a probability distribution (prior or posterior) on the model space requires that the model space is known—but this is not the case. It has been proposed to use uniform distribution over Bayesian networks, but this is incoherent because of multiple counts. Thus, as a first step towards a reasonable treatment of prior and posterior distributions, we must know the model space.

3. Especially for the assessment of appropriate prior distributions knowledge about the frequency of certain model classes in the set of all possible models is necessary. Invoking prior information can improve model learning substantially. We usually know a lot before we start selecting a complex model for a data set. We have selected the variables, e.g., such that we do not suppose the empty graph will be the best model. We know something about the number of components or the connectivity. Sometimes parts of a model are assumed to be fixed a priori leading to "structural zeros" in the adjacency matrix. There is a soft transition from "knowing for sure" (structural zeros) to "knowing nothing" (assigning equal probabilities).

4. For the theory of conditional independence models it is of principle interest to know more about properties that help to structure the model space. An excellent example is *model inclusion*. We will see below that there are more properties that are useful in evaluating and comparing models. An important distinction is that between unlabeled and labeled models. *Symmetries* will play an essential role in the proposed enumeration methods. *Boundary vertices*, a generalization of terminal nodes (as defined for directed graphs) to essential graphs (containing directed and undirected edges), will help us to order the models according to levels.

5. Of principle interest is the introduction of *representatives*. Representatives introduce a canonical standard order so that one and the same model is always visually drawn by the same patter or represented by the same adjacency matrix.

6. There are promising connections to certain types of *random graphs*. Monte Carlo methods in learning Bayesian networks were used [9].

7. Last not least, it is valuable to study the limiting behavior of various properties as the size of the graphs increases.

2 Essential graphs

Essential graphs (EGs) were introduced by Andersson, Madigan and Perlman [1]. An EG represents one and only one conditional independence model. Usually one EG corresponds to several Bayesian networks (BNs). The EG and the corresponding BNs do all have the same skeleton, that is, each vertex has the same neighbors, but some of the edges in the EG are undirected. There are three types of EGs: (i) acyclic directed EGs consisting of directed edges only, (ii) undirected EGs consisting of undirected edges only, and (iii) hybrids consisting of both, directed *and* undirected edges.

The most salient difference between BNs and EGs concerns the conditions in which a directed edge in a BN becomes an undirected edge in an EG. To define these conditions Andersson et al.[1] introduced the concepts of weakly and strongly protected arcs.

Definition 1 (Weakly protected arc) *Let $G(V, E)$ be a BN, $\{h, t, p, c\} \subseteq V$ four vertices in it, and $t \to h \in E$ be an arc from (tail) t to (head vertex) h. The arc $t \to h$ is weakly protected if one of the following conditions holds:*

Grandparent-parent condition *t has a parent p that is not a parent of h,*

$p \quad t \quad h$

Vee-condition *h has a parent p that is not a parent of t,*

$p \quad t \quad h$

Grandparent-grandchild condition *t and h have a common child c.*

$t \quad p \quad h$

For BNs only weakly protected arcs are relevant. In the case of EGs, though, one more condition involving undirected edges is relevant.

Definition 2 (Strongly protected arc) *The arc $t \to h$ is strongly protected if*

Weakly protected *the arc is weakly protected, and if*

Diagonal *t has two (undirected) siblings, s_1 and s_2, that both are parents of h, but are not mutually connected by an edge.*

It is also useful to introduce unprotected arcs:

Definition 3 (Weakly unprotected arc) *In a BN $t \to h$ is unprotected if*

Wedge *t has two unconnected children, h and c*

Moral child *t and h have a common child c.*

In these definitions the application of Shachter's theorem of arc reversal [13] for the special case of equal parents plays an important role. Shachter's theorem says that if an arc $a \to b$ is reversed so that $a \leftarrow b$, then all parents of b now also point to a and all parents of a point to b; that is, each node "inherits" its parents to the other one. The edged to these parents must be newly added to the graph, thus changing the skeleton of the graph. Arc reversal does not induce such additional new arcs if a and b must have the same parents (except the relation between a and b themselves). If two vertices have the same parents, then arc reversal does not change the probabilistic properties of the independence model.

3 Boundary vertices

In a directed graph a vertex without children is a terminal node or a sink. This concept is extended so that it can be applied to essential graphs. In an essential graph a *boundary vertex* is terminal in at least one of its equivalent BNs. If all undirected edges in an EG are replaced by directed arcs so that the EG is transformed into one of its equivalent BNs, then a vertex is a boundary vertex if there is at least one BN in which it is a terminal node.

A boundary vertex can be identified in a BN or in an EG with the help of the following definition:

Definition 4 *In a BN (in an EG) a vertex is a boundary vertex if it has no weakly (strongly) protected child.*

An isolated vertex having no neighbors is a boundary vertex. All vertices in a complete component of a BN (having only undirected edges in the corresponding EG) are boundary vertices.

Boundary vertices may easily be identified when the conditional independence model is represented by a perfect sequence [8]. In perfect sequences a boundary vertex is a vertex occurring only once in the whole sequence [7].

	0	1	2	...	...	$n-1$
0						
1	2^0					
2	2^1	2^2				
...	...	...	...	...		
$n-1$	...	...	...	...	$2^{\binom{n}{2}}$	

Table 1: Weights to order the vertices in an adjacency matrix. Low numbers correspond to high weights and *vice versa* so that the matrix will be maximized by 1s top-down and from left to right.

4 Layers

We use the boundary vertices to partially order the vertices of a BN (or an EG). For a given graph we first collect the set of all its boundary vertices an put them in layer 1. We next remove the vertices in this set from the graph and collect the boundary vertices of the remaining graph in layer 2. We repeat finding and removing sets of boundary vertices iteratively till the graph is empty.

The set of boundary vertex of an EG is called a *layer*. A sequence of layers $L_1, L_2, \ldots, L_m$ is obtained by first finding L_m, then removing of all vertices in L_m from EG, finding L_{m-1} of the remaining graph and repeating this until the remaining graph is empty. The sequence of layers induces a partial order on the vertices. Throughout we order the vertices layer-wise increasingly (from left to right), so that the vertices in L_1 are assigned ascending indexes $1, \ldots, |L_1|$ etc. At present the indexes within the layers are arbitrary. Below we will introduce an ordering also *within* the layers.

In an EG in which the vertices are ordered by layers all arcs point into one direction. We will draw all arcs from left to right. To represent the graph only the lower left half of the adjacency matrix is needed. The number of possible orderings of a BN is obtained by the product $|L_1|!\ |L_2|!\ |L_m|!$ The orderings may be represented in a tree. Each ordering corresponds to an adjacency matrix and is a topological sort of the graph. From now on we assume that the EG is ordered by layers and labeled from 1 to n [1].

To designate a canonical ordering of the vertices in the set of possible orderings a function is defined on the adjacency matrix that assigns a number to each 0-1-pattern in the adjacency matrix. It is convenient to use the decimal number that encodes the binary number that corresponds the cells of the "unfolded" matrix. This corresponds to taking the sum over a powers of 2 code. For one pattern of 0s and 1s the function obtains a maximum and the according pattern is used to define a *representative* graph. Enumerating representatives is equivalent to enumerating (unlabeled) EGs.

Traversing the tree of boundary vertices corresponds to working through the cross product of the permutations of the vertices in the various layers. One way

[1] or 0 to $n-1$ as we are used to C/++ style counting

to find representatives and to enumerate essential graphs is to generate (not necessarily all) within-layer permutations and to test for a maximum of the function. This is what we actually do to check algorithms for better methods—not very efficient but foolproof.

5 Symmetry

A main disadvantage of the method to enumerate EGs described in the previous section is that it is not unitive, it does not use a visually or otherwise easily identifiable property. Seeing many examples of representatives and non-representatives, we hit upon a criterion that is relatively easy to "see", namely symmetry. Symmetries in an EG allow to exchange vertices along with their adjacencies without changing the (unlabeled) structure.

Two vertices can change their position if they both belong to the same layer. Moreover, the adjacencies of the two vertices must map in a one-one way. Otherwise one vertex would have an adjacency with no corresponding adjacency in the second vertex. Thus, the two vertices must both have the same number of adjacencies in each of the layers. In previous work we incorrectly used this cardinality criterion—called the "signature" of a vertex [2]—to find identical structures. Unfortunately though, having the same signature is only a necessary but not a sufficient condition for the exchangeability of two vertices.

Definition 5 (Symmetry) *Two vertices X_i and X_j are symmetric, denoted by sym(X_i, X_j),*

1. *if they both are in the same layer and have the same number of neighbors in this layer, or*

2. *if their neighbors are pairwise symmetric.*

For a given EG we find the symmetric vertices iteratively. In the first step we set `sym(i,j) = true` if X_i and X_j have the same number of neighbors in the layer to which they both belong. If they are not in the same layer we set `sym(i,j) = false`. In the second step all pairs of vertices not having pairwise symmetric neighbors are set to `sym(i,j) = false`. The second step is repeated until the resulting matrix of pairwise symmetries does not change any more.

6 Enumerating essential graphs

Symmetry helps to find representatives and to find algorithms in many enumeration problems. An excellent treatment of this topic is given by Williamson [17, chapter 4]. We first turn to the problem of how many distinguishable labellings a given EG has. With the help of the symmetry classes the answer is easy.

[2] The signature of a vertex X_i in an EG with m layers $L_1, L_2, \ldots, L_m$ is a vector consisting of the number of adjacencies in each of the layers, $sig(X_i) = (|L_1|, |L_2|, \ldots, |L_m|.)$

We collect mutually symmetric vertices in *symmetry classes*. The cardinality of these classes is denoted by $\sigma_1, \sigma_2, \ldots, \sigma_q$. We obtain the following result:

Theorem 1 (Number of labelings of an EG) *The number of labellings of a given EG, G, with $n-1$ vertices, q symmetry classes with cardinalities $\sigma_1, \sigma_2, \ldots, \sigma_q$ is*

$$\lambda = \frac{n!}{\sigma_1!\ \sigma_2!\ \cdots \sigma_q!}\,.$$

If there are no symmetries, then there are $n!$ different labellings of n objects. If σ_1 of them are indistinguishable, then there are $\sigma_1!$ permutations that count only as one pattern so that there now are only $\frac{n!}{\sigma_1!}$ distinguishable labellings etc. for the remaining symmetry classes.

The enumeration of unlabeled EGs is more difficult. Symmetries will also be used to count unlabeled EGs. Switching the positions of two symmetric vertices is an operation that, while acting on an EG, leaves its shape invariant.

We follow the principle that of two symmetric vertices, the left vertex is always "served" first in the sense that it connects to a predecessor or a successor in the graph.

Now the following fact is obvious: In an EG with linearly ordered vertices $X_0, X_1, \ldots, X_{n-1}$ a vertex at position u may move to position i, $i < u$, without changing the structure of the graph left of position i if X_u and X_i have the *same* adjacencies in the subgraph left of position i. In a BN we would say, that X_u and X_i have the same parents. It is also obvious that in the general case *sameness* is replaced by *symmetry*.

This property is exploited to select a "maximal" linear order for the vertices of an EG. An order is maximal if, where ever possible, all edges are at their most left hand side position. This is equivalent to preferring adjacency matrices with 1s at the top and on left hand side positions, the weights following powers of 2 as shown in Table 1.

We denote non-adjacencies by $X_i \sqcup X_j$ and adjacencies (directed or undirected edges) by $X_u \rightharpoonup X_v$. Moreover, we say

1. edge (X_j, X_i) dominates edge (X_j, X_u), if $u < i$ and
2. edge (X_j, X_i) dominates edge (X_v, X_i), if $j < v$.

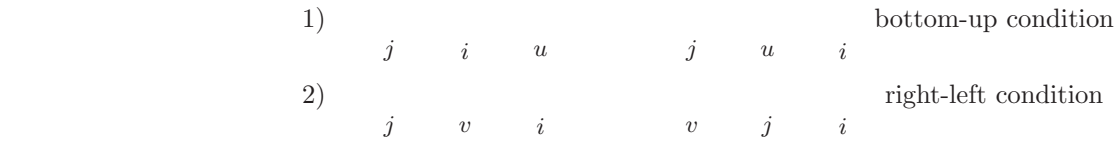

Figure 1: Left the two elementary non-maximal patterns; right the corresponding maximal patterns after moving symmetric vertices to the left hand side. Symmetric vertices have filled dots.

We introduce the following definition:

Definition 6 (Maximal EG) *An EG with linearly ordered vertices* $X_0, X_1, \ldots, X_{n-1}$ *is maximal if*

1. *there is no triple* (X_j, X_i, X_u) *with* $j < i < u$ *and* $X_j \sqcup X_i$ *and* $X_j \rightharpoonup X_u$, *such that sym*(X_i, X_u) *in the subgraph induced by* $X_0, X_1, \ldots, X_i, X_u$, *but without the edge* (X_j, X_u), $G^u \backslash (X_j, X_u)$, *and*

2. *there is no triple* (X_j, X_v, X_i) *with* $j < v < i$ *and* $X_j \sqcup X_i$ *and* $X_v \rightharpoonup X_i$, *and*

   (a) *sym*(X_v, X_j) *in the subgraph induced by* $(X_0, X_1, \ldots, X_j, X_v)$, *but without the edge* (X_v, X_i), $G^v \backslash (X_v, X_i)$, *and*

   (b) *there exists a subset in* $X_{v+1}, \ldots, X_{n-1}$ *such that to each vertex in* $X_{v+1}, \ldots, X_i$ *there corresponds a vertex in* $X_{v+1}, \ldots, X_{n-1}$ *that has the same or dominant edges in the subgraph induced by* $X_{v+1}, \ldots, X_i$.

The two conditions correspond to the two patterns shown in Figure 1, where the symmetric vertices are marked by filled dots. Condition 2a) ensures that there exists a permutation that keeps the structure between position v and i invariant or improves it while the structure left of v improves.

In the adjacency matrix we test the two conditions in the following way:

Bottom-up We build the subgraph G' by setting all rows and columns between positions i and u to 0, also all rows below u, and also cell (i, j). If if in G' sym(X_k, X_i), then the EG is not maximal.

Right-left We build a subgraph G' by setting all rows and columns between positions v and i to 0, also all rows below i, and also cell (i, j). If in G' sym(X_v, X_j) holds, then we first reset the adjacency matrix and then check whether there is a 1-1 mapping from the rows between position v and i to a subset of rows below v when X_i and X_j switch their positions. If there exists such a mapping, then the EG is not maximal.

The second condition looks more complicated than it (computationally) is.

Usually several vertices are required that successively switch their positions to obtain the maximal representative—corresponding to permutations of the vertices with several inversions. In a non-maximal EG, though, there must always be at least one inversion that moves the start or the end of an edge to the left hand side and this inversion corresponds either to a bottom-up or right-left move in the adjacency matrix. The procedure described only rejects non-maximal structures. To perform all necessary permutational steps to transform a given EG into its representative requires to apply the procedure recursively.

We give two examples how to test whether a given EG is maximal. The first one explains the bottom-up condition. Figure 3 shows a non-maximal graph. Vertex 4 and vertex 6 are symmetric in the subgraph induced by their predecessors (Figure 3). This subgraph is obtained by overwriting row 5 by

	0	1	2	3	4	5
0						
1	0					
2	0	0				
3	0	0	0			
4	1	1	0	0		
5	0	1	1	1	0	

Table 2: Adjacency matrix of a non-maximal EG. If row 4 & 5 and columns 2 & 3) switch positions the structure becomes maximal

row 6 and deleting the row 6. The edge (1, 6) may therefore be changed to (1, 4). Note that the resulting structure would require more changes to become maximal. The maximal permutation is (1, 0, 2, 3, 6, 4, 5). To reject the original structure, though, one violation of the maximality principle is sufficient.

As a second example, Table 4 shows an adjacency matrix in which the 0 in cell (5, 0) can be replaced by moving the 1 in cell (5, 2) to the left. This leaves the entries in the part above row 5 unchanged. Figure 4 shows the subgraph induced by the predecessors of vertex 5. Note that it does not contain the critical edge (2, 4). To test whether there is a "better" position for this edge we have to omit it out for a moment. The maximal permutation for this example is (2, 3, 0, 1, 4, 5, 6). Again, one violation of the maximum condition is sufficient to reject the structure.

We state the following two consequences:

1. The number of unlabeled EGs is equal to the number of maximal EGs.

2. The total number of labeled EGs with a given number of vertices is equal to the sum of the number of labellings of all unlabeled structures.

0 1 2 3 4 5

Figure 2: Essential graph associated with the adjacency matrix of Tabel 2. The arc from 3 to 5 is not maximal and should connect 2 and 4 to make the graph maximal

	0	1	2	3	4	5	6
0							
1	0						
2	1	1					
3	0	0	0				
4	1	0	**0**	0			
5	1	0	0	0	0		
6	0	1	**1**	1	0	0	

Table 3: Adjacency matrix of a non-maximal EG, bottom-up condition. When vertex 6 moves to position 4 row 4 gets a higher evaluation while leaving rows 0, 1, 2, and 3 unchanged; the 0 in cell (4, 2) is then replaced by the 1 from cell (6, 2).

0 1 2 3 4 5 6

Figure 3: Subgraph induced by the predecessors of vertex 4 and 6 of the example shown in Table 3. The symmetry of 4 and 6 is obvious.

	0	1	2	3	4	5	6
0							
1	1						
2	0	0					
3	0	0	1				
4	1	0	1	0			
5	**0**	1	**1**	0	0		
6	0	0	0	1	0	0	

Table 4: Adjacency matrix of a non-maximal EG, right-left condition. When vertex 2 moves to position 0 row 5 gets a higher evaluation while leaving rows 0, 1, 2, 3, and 4 unchanged. The 0 in cell (5, 0) is replaced by the 1 from cell (5, 2):

0 1 2 3 4 5 6

Figure 4: Subgraph induced by the predecessors of vertex 0 and 4 of the example shown in Table 4. The symmetry of 0 and 2 is obvious.

7 Discussion

At present we are writing a computer program that generates and enumerates maximal EGs. We are exploiting the fact that maximal structures must always be extensions of smaller maximal structures. It is thus possible to generate EGs that grow vertex by vertex top-down in the adjacency matrix. We count the number of different structures for essential graphs with one component only. It cannot be excluded that with the help of boundary vertices and symmetry formulas may be found that allow to calculate cardinalities. The property of boundary vertices, layers, and symmetry are easily translated into representations by perfect sequences. Perfect sequences may be more efficient for algorithms than adjacency matrices.

The representation of Bayesian networks and essential graphs by their canonical form offers the possibility of standardizing visual representations. In standardized representations two different pictures depict two different models.

Our ultimate motivation is to find a method that—from a Bayesian perspective of learning model structures—allows a satisfactory treatment of probability distributions over sets of structures. In the literature it has repeatedly been pointed out [4] that the number of models is so huge that a straightforward statistical Bayesian approach to model learning is prohibited. Structuring the model space hierarchically and working with conditional prior or posterior distributions may help to overcome some of these difficulties.

There are fields in which very complex models are inferred from the data, but where there are chronic difficulties to replicate the inferred models. Examples can be found in behavioral genetics and in fMRI-studies. Low model reliability may be a consequence of complexity. If there is a conflict between complexity on the one hand and sound inference methods, we prefer simple models and sound inference. Improved methods for learning conditional independence models may help to overcome some of these difficulties. Being able to evaluate and compare the probability of models may avoid over-fitting and the pitfalls of non-holistic analysis.

References

[1] Andersson, S. A., Madigan, D., and Perlman, M. D. (1998). A characterization of Markov equivalence classes for acyclic digraphs. Annals of Statistics, **25**, 505-541.

[2] Gillispie, S. B. and Perlman, M. D. (2001). Enumerating Markov equivalence classes of acyclic digraph models. *Uncertainty in Artificial Intelligence: Proc. of the Seventeenth Conf.*. Morgen Kaufmann, San Francisco, 171-177.

[3] Gillispie, S. and Perlman, M. D. (2002). The size distribution for Markov equivalence classes of acyclic digraph models. *Artificial Intelligence*, **141**, 137-155.

[4] Heckerman, D. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, **20**, 197-243.

[5] Jiroušek, R. (1998). Graph Modelling without graphs. In: Bouchon-Meunier, B. and Yager, R. R. (Eds.), Proc. of the 7th Int. Conf. on Information processing and Management of Uncertainty in Knowledge-based Systems, Editions E. D. K. Paris, pp. 806-816.

[6] Jiroušek, R. (2000). Marginalization in composed probabilistic model, in *Proceedings of the 13th Conference in Uncertainty in Artificial Intelligence UAI'97*, D. Geiger and P. P. Shenoy (Eds.), Morgan Kaufmann, San Francisco, 1997, pp. 274-281.

[7] Jiroušek, R. (2001). Detection of independence relations from persegrams, in 4^{th} *Czech-Japan Seminar on Data Analysis and Decision Making under Uncertainty*, R. Jiroušek and J. Vejnarová (Eds.), Jindřichøuv Hradec, Czech Republic, September 14-17, pp.52-62.

[8] Jiroušek, R. (2002). Decomposition of multidimensional distributions represented by perfect sequences. *Annals of Mathematics and Artificial Intelligence*, **35**, 215-226.

[9] Kleiter, G. D. (1999). The posterior probability of Bayes nets with sgrong dependences. *Soft Computing*, **3**, 162-173.

[10] Kleiter, G. D. (2003). Enumerating essential graphs. In R. Jiroušek and J. Vejnarová (Eds.) Proceedings of the 6^th Workshop on Uncertainty Processing, Hejnice (Czech Republic), September 24-27, pp. 135-148.

[11] Kleiter, G. D. (2004). Enumerating equivalence classes of Bayesian networks by perfect sequences. IPMU 2004, Perugia, in print.

[12] Robinson, R. W. (1976). Counting unlabeled acyclic digraphs. In *Proceedings of the Fifth Australian Conference on Combinatorial Mathematics*, Melbourne, Australia, 28-43.

[13] Shachter, R. D. (1986). Evaluating influence diagrams. *Operations Research*, **34**, 871-882.

[14] Steinsky, B. (2003) Enumeration of labelled chain graphs and labelled essential directed acyclic graphs. *Discrete Mathematics*, **270**, 267-278.

[15] Steinsky, B. (2003) Efficient coding of labeled directed acyclic graphs. *Soft computing*, **7**, 350-356.

[16] Volf, M. and Studený, M. (1999). A graphical characterization of the largest chain graphs. *International Journal of Approximate Reasoning*, **20**, 209-236.

[17] Williamson, S. G. (1985). *Combinatorics for Computer Science*. Computer Science Press, Rockville, ML.